A Micro-benchmark Suite for Evaluating Hadoop MapReduce on High-Performance Networks

Dipti Shankar(✉), Xiaoyi Lu, Md. Wasi-ur-Rahman, Nusrat Islam, and Dhabaleswar K. (DK) Panda

Department of Computer Science and Engineering, The Ohio State University, Columbus, USA
{shankard,luxi,rahmanmd,islamn,panda}@cse.ohio-state.edu

Abstract. Hadoop MapReduce is increasingly being used by many datacenters (e.g. Facebook, Yahoo!) because of its simplicity, productivity, scalability, and fault tolerance. For MapReduce applications, achieving low job execution time is critical. Since a majority of the existing clusters today are equipped with modern, high-speed interconnects such as InfiniBand and 10 GigE, that offer high bandwidth and low communication latency, it is essential to study the impact of network configuration on the communication patterns of the MapReduce job. However, a standardized benchmark suite that focuses on helping users evaluate the performance of the stand-alone Hadoop MapReduce component is not available in the current Apache Hadoop community. In this paper, we propose a micro-benchmark suite that can be used to evaluate the performance of stand-alone Hadoop MapReduce, with different intermediate data distribution patterns, varied key/value sizes, and data types. We also show how this micro-benchmark suite can be used to evaluate the performance of Hadoop MapReduce over different networks/protocols and parameter configurations on modern clusters. The micro-benchmark suite is designed to be compatible with both Hadoop 1.x and Hadoop 2.x.

Keywords: Big data · Hadoop MapReduce · Micro-benchmarks · High-performance networks

1 Introduction

MapReduce, proposed by Google [8], has been seen as a viable model for processing petabytes of data. The Apache Hadoop project [23], an open-source implementation of the MapReduce computing model, has gained widespread acceptance and is widely used in many organizations around the world. MapReduce is extensively

This research is supported in part by National Science Foundation grants #OCI-1148371, #CCF-1213084 and #OCI-1347189. It used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number #OCI-1053575.

DOI: 10.1007/978-3-319-13021-7_2

adopted by various applications to perform massive data analysis and is hence required to deliver high performance. While Hadoop does attempt to minimize the movement of data in the network, there are times when MapReduce does generate considerable network traffic, especially during the intermediate data shuffling phase [17], which is communication intensive.

Several modern, high-speed interconnects such as InfiniBand and 10 GigE, are used widely in clusters today. The data shuffling phase of the MapReduce job can immensely benefit from the high bandwidth and low latency communication offered by these high-performance interconnects. In order to evaluate this improvement potential, we require benchmarks that can give us insights into the factors that affect MapReduce as an independent component. The performance of Hadoop MapReduce is influenced by many factors such as network configuration of the cluster, controllable parameters in software (e.g. number of maps/reduces, data distribution), data types, and so on. To get optimal performance, it is necessary to tune and optimize these factors, based on cluster and workload characteristics. Adopting a standardized performance benchmark suite to evaluate these performance metrics in different configurations would be good for Hadoop users. For Hadoop developers, a benchmark suite with these capabilities could help evaluate the performance of new MapReduce designs.

At present, we lack a standardized benchmark suite that focuses on helping users evaluate the performance of the Hadoop MapReduce as a stand-alone component. Current, commonly used benchmarks in Hadoop, such as Sort and TeraSort, usually require the involvement of HDFS. The performance of the HDFS component has significant impact on the overall performance of the MapReduce job, and this interferes in the evaluation of the performance benefits of new designs for MapReduce. Furthermore, these benchmarks do not provision us to study the impact of changing data distribution patterns, varying data types, etc., on the performance of the MapReduce job. Such capabilities are very useful for optimizing the parameters and the internal designs of Hadoop MapReduce. With this as background, the basic motivation of this paper is: *Can we design a simple micro-benchmark suite to let users and developers in the Big Data community evaluate, understand, and optimize the performance of Hadoop MapReduce in a stand-alone manner over different networks/protocols?*

In this paper, we propose a comprehensive micro-benchmark suite to evaluate the performance of stand-alone Hadoop MapReduce. We provide options for varying different benchmark-level parameters such as intermediate data distribution pattern, key/value size, data type, etc. Our micro-benchmark suite can also dynamically set the Hadoop MapReduce configuration parameters, like number of map and reduce tasks, etc. We display the configuration parameters and resource utilization statistics for each test, along with the final job execution time, as the micro-benchmark output.

This paper makes the following key contributions:

1. Designing a micro-benchmark suite to evaluate the performance of stand-alone Hadoop MapReduce, when run over different types of high-performance networks.

2. A set of micro-benchmarks to measure the job execution time of Hadoop MapReduce with different intermediate data distribution patterns.
3. Illustration of the performance results of Hadoop MapReduce using these micro-benchmarks over different networks/protocols (1 GigE/10 GigE/IPoIB QDR (32 Gbps)/IPoIB FDR (56 Gbps)).
4. A case study on enhancing Hadoop MapReduce design by using RDMA over native InfiniBand, undertaken with the help of the proposed micro-benchmark suite.

The rest of the paper is organized as follows. In Sect. 2, we discuss related work in the field. We present design considerations for our micro-benchmark suite in Sect. 3 and the micro-benchmarks in Sect. 4. In Sect. 5, we present the results of performance tests, obtained with our micro-benchmark suite. Section 6 shows a case study with RDMA-enhanced MapReduce. Finally, we conclude the paper in Sect. 7.

2 Related Work

Over the years, many benchmarks have been introduced in the areas of Cloud Computing and Big Data. MRBench [13] provides micro-benchmarks in the form of MapReduce jobs of TPC-H [4]. MRBS [21] is a benchmark suite that evaluates the dependability of MapReduce systems. It provides five benchmarks for several application domains and a wide range of execution scenarios. Similarly, HiBench [9] has extended the DFSIO program to compute the aggregated throughput by disabling the speculative execution of the MapReduce framework. It also evaluates Hadoop in terms of system resource utilization (e.g. CPU, memory). MalStone [6] is a benchmark suite designed to measure the performance of cloud computing middleware when building data mining models. Yahoo! Cloud Serving Benchmark (YCSB) [7] is a set of benchmarks for performance evaluations of key/value-pair and cloud data-serving systems. YCSB++ [18] further extends YCSB to improve performance understanding and debugging. BigDataBench [25], a benchmark suite for Big Data Computing, covers typical Internet service workloads and provides representative data sets and data generation tools. It also provides different implementations for various Big Data processing systems [1,14].

In addition to the above benchmarks that address the Hadoop framework as a whole, micro-benchmark suites have been designed to study some of its individual components. The micro-benchmark suite designed in [10] helps with detailed profiling and performance characterization of various HDFS operations. Likewise, the micro-benchmark suite designed in [16] provides detailed profiling and performance characterization of Hadoop RPC over different high-performance networks. Along these lines, our proposed micro-benchmark suite introduces a performance evaluation tool for stand-alone Hadoop MapReduce, that does not need HDFS or any other distributed file system.

3 Design Considerations

The performance of a MapReduce job is usually measured by its execution time. It can be significantly influenced by numerous factors such as the underlying network or the communication protocol, number of map tasks and reduce tasks, intermediate shuffle data pattern, and the shuffle data size, as illustrated in Fig. 1(a).

Essentially, the efficiency of the network intensive data shuffling phase is determined by how fast the map outputs are shuffled. Based on these aspects, we consider the following dimensions to design the Hadoop MapReduce micro-benchmark suite,

Intermediate data distribution: Map tasks transform input key/value pairs to a set of intermediate key/value pairs. These intermediate key/value pairs are shuffled from the mappers, where they are created, to the reducers where they are consumed. Depending upon the MapReduce job, the distribution of intermediate map output records can be even or skewed. A uniformly balanced load can significantly shorten the total run time by enabling all reducers to finish at about the same time. In jobs with a skewed load, some reducers complete the job quickly, while others take much longer, as the latter have to process a more sizeable portion of the work. Since it is vital for performance to understand whether these distributions can be significantly impacted by the underlying network protocol, we consider this an important aspect of our micro-benchmark suite.

Size and number of key/value pairs: For a given data size, the size of the key/value pair determines the number of times the Mapper and Partitioner functions are called; and, in turn, the number of intermediate records being shuffled. Our micro-benchmark suite provides support for three related parameters: key size, value size and number of key/value pairs. Through these parameters, we can specify the total amount of data to be processed by each map, amounting to the total shuffle data size. These parameters can help us understand how the nature of intermediate data, such as the key/value pair sizes, can impact the performance of the MapReduce job, on different networks.

Number of map and reduce tasks: The number of map and reduce tasks is probably the most basic Hadoop MapReduce parameters. Tuning the number of map and reduce tasks for a job is essential for optimal performance and hence we provide support to vary the number of map and reduce tasks in our micro-benchmark suite.

Data types: Hadoop can process many different types of data formats, from flat text files to databases. Binary data types usually take up less space than textual data. Since disk I/O and network transfer will become bottlenecks in large jobs, reducing the sheer number of bytes taken up by the intermediate data can provide a substantial performance gain. Thus, data types can have a considerable impact on the performance of the MapReduce job. Our micro-benchmark suite is designed to support different data types.

Network configuration: The intermediate data shuffling phase, which is the heart of MapReduce, results in a global, all-to-all communication. This accounts for a rather significant amount of network traffic within the cluster, although it varies from job to job. This is an important consideration, especially when expanding the cluster. Hence, it is essential to compare and contrast the impact that different network interconnects/protocols have on the performance of the MapReduce job. Our micro-benchmark suite is capable of running over any network and cluster configuration.

Resource utilization: The multi-phase parallel model of MapReduce and its scheduling policies have a significant impact on various systems resources such as the CPU, the memory, and the network, especially with an increasing number of maps and reduce tasks being scheduled. As it is essential to understand the correlation between network characteristics and resource utilization, our micro-benchmark suite provides the capability to measure the resource utilization, during the course of the MapReduce job.

4 Micro-benchmarks for Hadoop MapReduce

In this section, we present the overall design of the micro-benchmark suite, and describe the micro-benchmarks implemented based the various design factors outlined in Sect. 3.

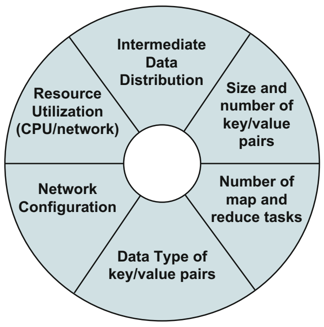
4.1 Overview of Overall Micro-benchmark Suite Design

In this study, we develop a micro-benchmark suite for stand-alone Hadoop MapReduce, in order to provide an understanding of the impact of different factors described in Sect. 3 on the performance of the MapReduce job when run over different networks. As illustrated in Fig. 1(b), these micro-benchmarks have the following key features:

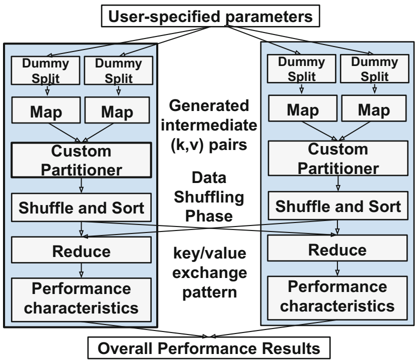
Stand-alone feature: Each micro-benchmark is basically a MapReduce job that is launched without HDFS or any other distributed file system. In order to achieve this,

(1) For the Map phase, we define a custom input format, namely, `NullInputFormat`, for the mapper instances. This empty input format creates dummy input splits based on the number of map tasks specified, with a single record in each. Each map task generates a user-specified number of key/value pairs in memory, and passes it on as map output.

(2) For the Reduce phase, we make use of `NullOutputFormat` [3], defined in the MapReduce API, as the output format. Each reduce task aggregates intermediate data from the map phase, iterates over them and discards it to `/dev/null`. This is ideal for our micro-benchmarks, since we evaluate MapReduce as a stand-alone component.


Intermediate Data Distribution
Size and number of key/value pairs
Number of map and reduce tasks
Data Type of key/value pairs
Network Configuration
Resource Utilization (CPU/network)

(a) Design considerations


User-specified parameters
Dummy Split
Dummy Split
Map
Map
Custom Partitioner
Shuffle and Sort
Reduce
Performance characteristics
Generated intermediate (k,v) pairs
Data Shuffling Phase
key/value exchange pattern
Dummy Split
Dummy Split
Map
Map
Custom Partitioner
Shuffle and Sort
Reduce
Performance characteristics
Overall Performance Results

(b) Micro-benchmark Design

Fig. 1. Design overview of the MapReduce micro-benchmarks

Custom Partitioners: The partitioning phase that takes place after the map phase and before the reduce phase, partitions the data gets across the reducers according to the partitioning function. To simulate different intermediate data distribution scenarios, we employ different custom partitioners. We support three different intermediate data distribution patterns, namely, average distribution, random distribution and skewed distribution. These three distributions cover most of the intermediate data distribution patterns found in MapReduce applications.

Configurable parameters: In our micro-benchmark suite, we provide parameters to vary the number of mappers and reducers. The user can specify the size in bytes and the number of the key/value pairs to be generated based on the intermediate shuffle data size of the MapReduce job. We also provide a parameter to indicate data type, such as `BytesWritable` or `Text`. Based on these parameters, the micro-benchmark suite generates the data to be processed by each map.

4.2 Micro-benchmarks

Using the framework described in Sect. 4.1 as basis, we define three micro-benchmarks. For each of these, we can vary the size and number of key/value pairs, to generate different sizes of shuffle data. These micro-benchmarks use the `Map` function to create specified number of key/value pairs. To avoid any additional overhead, we restrict the number of unique pairs generated to the number of reducers specified.

Average-distribution (MR-AVG): In this micro-benchmark, we distribute the intermediate key/value pairs uniformly amongst all of the reduce tasks. The custom partitioner defined for MR-AVG distributes the key/value pairs amongst the reducers in a round-robin fashion, making sure each reducer gets the same number of intermediate key/value pairs. This helps us obtain a fair comparison of the performance of a MapReduce job on different networks, when the intermediate data is evenly distributed.

Random-distribution (MR-RAND): In this micro-benchmark, we randomly distribute the intermediate key/value pairs among the reduce tasks. The custom partitioner defined for MR-RAND randomly picks a reducer and assigns the key/value pair to it. With the help of Java's Random class, the reducer is pseudo-randomly chosen, with its range specified as the number of reducers. With this limited range, the micro-benchmark more or less generates the same pattern of reducers, making sure each run gets similar intermediate key/value pairs to reducers mapping. This mapping is relatively close to an even distribution and thus helps us picture a fairly accurate comparison of the performance of a MapReduce job on different networks, when the intermediate data is randomly distributed among the reducers.

Skew-distribution (MR-SKEW): In this micro-benchmark, we distribute the intermediate key/value pairs unevenly among the reducers. Based on the number of reducers specified, the custom partitioner defined for MR-SKEW distributes 50 % of the intermediate key/value pairs to the first reducer, 25 % of the remainder to the second reducer, 12.5 % of the remaining to the third, and then randomly distributes the rest. Since this skewed distribution pattern is fixed for all runs, irrespective of the key/value pairs generated, we can guarantee a fair comparison on homogenous systems. This micro-benchmark helps us gain essential insights into the performance of MapReduce jobs with skewed loads, running over different network types. By determining the overhead of running a skewed load, we can determine if it is worthwhile to find alternative techniques that can mitigate load imbalances in Hadoop applications.

5 Performance Evaluation

5.1 Experimental Setup

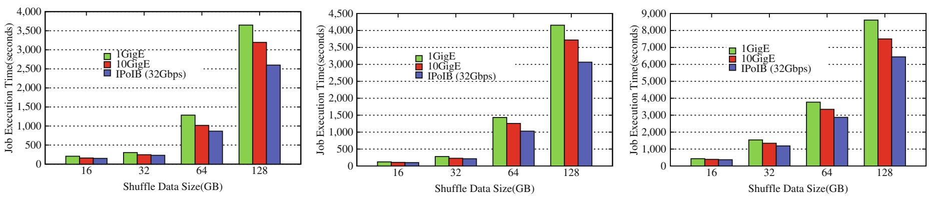
(1) Intel Westmere Cluster (Cluster A): This cluster has nine nodes. Each node has Xeon Dual quad-core processor operating at 2.67 GHz. Each node is equipped with 24 GB and two 1TB HDDs. Nodes in this cluster also have NetEffect NE020 10Gb Accelerated Ethernet Adapter that are connected using a 24 port Fulcrum Focalpoint switch. The nodes are also interconnected with a Mellanox switch. Each node runs Red Hat Enterprise Linux Server release 6.1.

(2) TACC Stampede [22] (Cluster B): We use the Stampede supercomputing system at TACC [22] for our experiments. According to TOP500 [24] list in June 2014, this cluster is listed as the 7th fastest supercomputer worldwide. Each node in this cluster is dual socket containing Intel Sandy Bridge (E5-2680) dual octa-core processors, running at 2.70GHz. It has 32 GB of memory, a SE10P (B0-KNC) co-processor and a Mellanox IB FDR MT4099 HCA. The host processors are running CentOS release 6.3 (Final). Each node is equipped with a single 80 GB HDD.

For Cluster A, we show performance comparisons over 1 GigE, 10 GigE, and IPoIB (32 Gbps). We evaluate our micro-benchmarks with Apache Hadoop 1.2.1 and JDK version 1.7. We also present some results with Apache Hadoop NextGen MapReduce (YARN) [5] version 2.4.1. On Cluster B, we compare IPoIB FDR (56 Gbps) and RDMA FDR (56 Gbps), with Apache Hadoop 1.2.1 running over IPoIB and RDMA for Apache Hadoop (v0.9.9) [2]. These results are presented in Sect. 6.

5.2 Evaluation along Different Dimensions

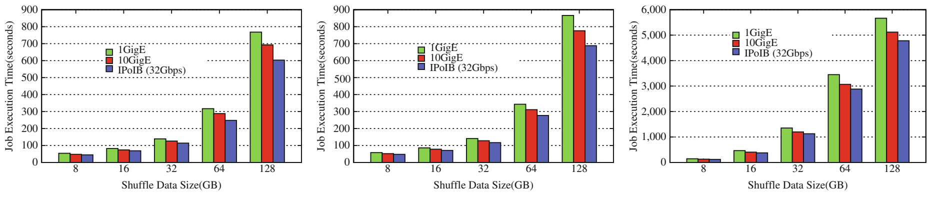
Evaluating impact of intermediate data distribution patterns: In this section, we present performance results obtained using micro-benchmarks presented in Sect. 4.2, to evaluate the impact of intermediate data distribution patterns on the MapReduce job execution time. We perform these tests on Cluster A, using `BytesWritable` data type and a fixed key/value pair size of 1 KB, with 16 map tasks and 8 reduce tasks on 4 slave nodes. We compare the performance with different shuffle data sizes, by varying the number of intermediate key/value pairs generated. Figure 2 shows comparison for job execution time with different intermediate data distribution patterns. From Fig. 2(a), it is clear that the job execution time for MR-AVG micro-benchmark decreases around 17 %, if the underlying interconnect is changed to 10 GigE from 1 GigE, and up to 24 %, when changed to IPoIB (32 Gbps). Similarly, from Fig. 2(b), job execution time for MR-RAND micro-benchmark decreases around 16 %, if the underlying interconnect is 10 GigE instead of 1 GigE, and up to 22 %, if it is IPoIB (32 Gbps). We observe that IPoIB (32 Gbps) improves the performance by about 8–10 %, as compared to 10 GigE, for both MR-AVG and MR-RAND micro-benchmarks. From Fig. 2(c), we can observe that the performance improves by about 11 % for MR-SKEW micro-benchmark, if we switch from 1 GigE from 10 GigE. Also, IPoIB (32 Gbps) performs better than 10 GigE, by about 12 %, as intermediate shuffle data sizes are scaled up. It can be observed that IPoIB (32 Gbps) provides better improvement with increased shuffle data sizes and more skewed workloads. Also, the skewed data distribution seems to double the job execution time for a given data size, as compared to the average distribution, irrespective of the underlying network interconnect.



(a) MR-AVG (b) MR-RAND (c) MR-SKEW

Fig. 2. Job Execution Time for different data distribution patterns on Cluster A

Evaluating with Apache Hadoop NextGen MapReduce (YARN): In this section, we evaluate the probable influence that the intermediate data distribution pattern has on the MapReduce job execution time for the Hadoop YARN Architecture [5], using Apache Hadoop 2.4.1. We perform these tests on Cluster A, with a key/value pair size of 1 KB, using 32map tasks and 16 reduce tasks on 8 slave nodes. We compare the performance with different shuffle data sizes, by varying the number of intermediate key/value pairs generated. From Fig. 3(a), it is clear that the job execution time for MR-AVG decreases around 11 %, if the underlying interconnect is changed to 10 GigE from 1 GigE, and by about 18 %, when changed to IPoIB (32 Gbps). Similarly, from Fig. 3(b), job execution time for MR-RAND micro-benchmark decreases around 10 %, when we switch from 1 GigE to 10 GigE, and up to 17 % improvement, when changed to IPoIB (32 Gbps). For MR-SKEW micro-benchmark, Fig. 3(c) shows that the performance of the MapReduce job improves by about 10–12 % with the use of high-speed interconnects. It can be observed that, IPoIB (32 Gbps) improves performance by about 7–10 % for all three micro-benchmarks, as compared to 10 GigE, with increased shuffle data sizes. Also, for a given data size, the skewed data distribution increases the job execution time by more than 3X, when compared to the average data distribution, irrespective of the underlying network interconnect. From Figs. 2 and 3, we can infer that, increasing cluster size and concurrency significantly benefits average and random data distribution patterns. From Figs. 2(c) and 3(c), it also evident that, even though the Map phase may benefit from the increased concurrency and cluster size, the Reduce phase of the MapReduce job with a skewed intermediate data distribution still depends on the slowest reduce task, and hence, the improvement is not as much.

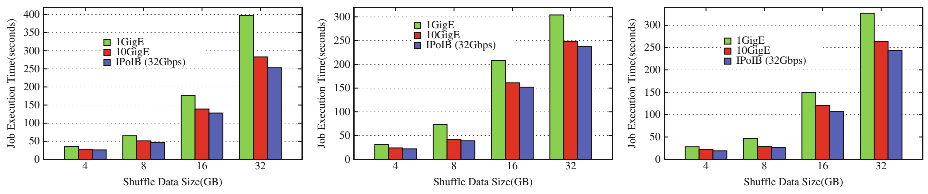


(a) MR-AVG (b) MR-RAND (c) MR-SKEW

Fig. 3. Job Execution Time with different patterns for YARN architecture on Cluster A

Evaluating impact of varying key/value pair sizes: In this section, we present results using MR-AVG micro-benchmark, to portray the impact of varying key/value pair size on the performance of the MapReduce job. We run these evaluations on Cluster A, with 16 map tasks and 8 reduce tasks on 4 slave nodes, for `BytesWritable` data type. Figure 4 shows job execution time comparisons with MR-AVG micro-benchmark for different key/value pair sizes. From Fig. 4(a), we can see that the job execution time for a key/value pair size of 100 bytes, decreases around 18 %, if the underlying interconnect is changed from

1 GigE to 10 GigE, and by about 22 %, when changed to IPoIB (32 Gbps). Figures 4(b) and (c) show similar performance improvement for key/value pair sizes of 1 KB and 100 KB, when underlying interconnect is changed to from 1 GigE to IPoIB (32 Gbps) and 10 GigE. We also observe that IPoIB (32 Gbps) performs slightly better than 10 GigE, by about 7–10 %, for all three key/value pair sizes. It can be seen that increasing the key/value pair sizes brings about lower job execution times for a given shuffle data size. For instance, the job execution time for 16 GB shuffle data size reduces from 128 to 107 s for IPoIB (32 Gbps) when key/value sizes are increased from 100 bytes to 10 KB. We can therefore infer that the size and number of key/value pairs can influence the performance of the MapReduce job running on different networks.



(a) MR-AVG with key/value size of 100 bytes

(b) MR-AVG with key/value size of 1 KB

(c) MR-AVG with key/value size of 10 KB

Fig. 4. Job Execution Time with MR-AVG for different key/value pair sizes on Cluster A

Evaluating impact of varying the number of map and reduce tasks: In this section, we present performance results with varying number of map and reduce tasks, using MR-AVG micro-benchmark, over 10 GigE and IPoIB (32 Gbps). We run these performance evaluations on Cluster A, with a key/value pair size of 1 KB. We vary the number of key/value pairs to generate different shuffle data sizes. In Fig. 5, we present performance evaluations with 8 map and 4 reduce tasks (8M-4R), and 4 map and 2 reduce tasks (4M-2R). For both these cases, Fig. 5 clearly shows that IPoIB (32 Gbps) outperforms 10 GigE, by about 13 %. It is evident that IPoIB (32 Gbps) gives better performance improvement with increased concurrency, as compared to 10 GigE. For instance, increasing the number of map and reduce tasks improved the performance of the MapReduce job by about 32 % for IPoIB (32 Gbps), while it improved by only 24 % for 10 GigE, for a shuffle data size of 32 GB. It can therefore be inferred that varying the number of map and reduce tasks can impact the load on the network.

Evaluating impact of data types: In this section, we present results of experiments done with the MR-RANDOM micro-benchmark, to understand the impact of data types on the performance of the MapReduce job over different networks interconnects. We run these experiments on Cluster A, using 16 map tasks and 8 reduce tasks on 4 slave nodes, with fixed key/value pair size of 1 KB,

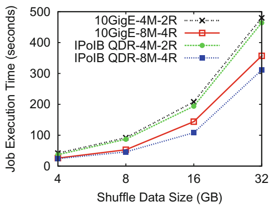
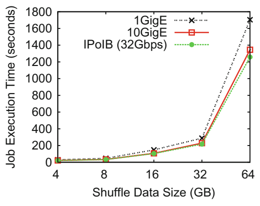
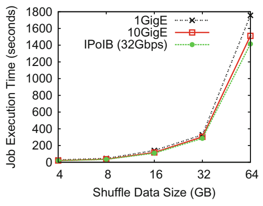


Fig. 5. Job Execution Time with varying number of maps and reduces on Cluster A



(a) Job Execution Time with `BytesWritable`



(b) Job Execution Time with `Text`

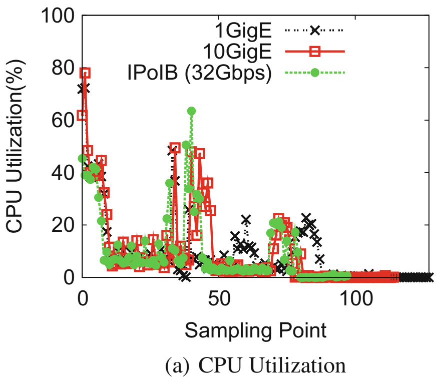
Fig. 6. Job Execution Time with `BytesWritable` and `Text` on Cluster A

to study `BytesWritable` and `Text` data types. Figure 6(a) shows the trend for `BytesWritable` and Fig. 6(b) shows the trend for `Text`, as we scale up to 64 GB. We observe that the job execution time decreases around 17–23 %, if the underlying interconnect is 10 GigE, instead of 1 GigE, and up to 28 % improvement, if it is IPoIB (32 Gbps). Also, IPoIB (32 Gbps) gives noticeable performance improvement over 10 GigE. It is evident that high-speed interconnects provide similar improvement potential to both data types. We plan to investigate other data types, as the next step.

Resource Utilization: In this section, we present results of experiments done with the MR-AVG micro-benchmark, to study the resource utilization patterns of MapReduce jobs over different network interconnects. We use 16 map tasks and 8 reduce tasks on 4 slave nodes for these experiments, on Cluster A. We present CPU and network utilization statistics of one of the slave nodes. Figure 7(a) shows the CPU utilization trends for MR-AVG benchmark, run with intermediate data size of 16 GB, a fixed key/value pair size of 1 KB and `BytesWritable` data type. It can be observed that CPU utilization trends of 10 GigE and IPoIB (32 Gbps) are similar to that of 1 GigE. Figure 7(b) shows the network throughput for the same. For network throughput, we consider the total number of megabytes received per second. IPoIB (32 Gbps) achieves a peak bandwidth of 950 MB/s, 10 GigE peaks at 520 MB/s, and 1 GigE peaks at 101 MB/s. These trends suggest that IPoIB (32 Gbps) makes better use of the resources, especially the network bandwidth, as compared to 10 GigE and 1 GigE.

6 A Case Study: Enhanced Hadoop MapReduce Design over Native InfiniBand

In previous research [19,20], we have designed and implemented a high-performance Hadoop MapReduce framework with RDMA over native InfiniBand, known as MRoIB. This is publicly available as a part of the RDMA for Apache Hadoop project (v0.9.9) [2,11,12,15,19,20]. We found that the stand-alone Hadoop MapReduce micro-benchmark suite proposed in this paper is



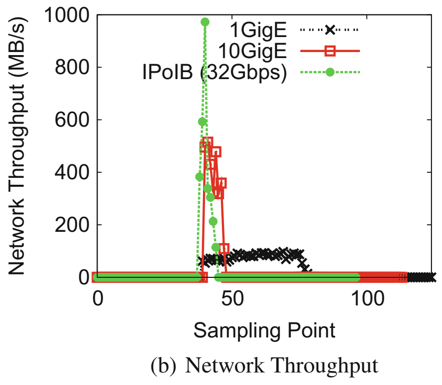
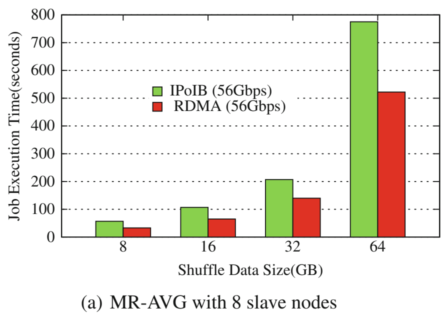


Fig. 7. Resource Utilization on one slave node for MR-AVG on Cluster A

extremely helpful in evaluating the performance of alternative MapReduce designs such as MRoIB, and in tuning different internal parameters to obtain optimal performance.



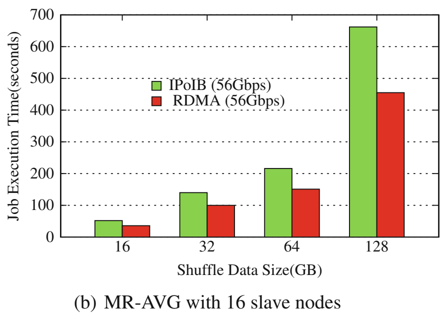


Fig. 8. Performance with different patterns for IPoIB Vs. RDMA on Cluster B (56 Gbps FDR)

Figure 8 illustrates the performance improvement possible with native IB as compared to IPoIB (56 Gbps) on Cluster B. We use `BytesWritable` data type and a fixed key/value pair size of 1 KB, with 32 map tasks and 16 reduce tasks. We vary the number of key/value pairs to generate different shuffle data sizes, and study the MR-AVG micro-benchmark. We omit the other two micro-benchmarks due to space constraints. From Fig. 8(a), we observe that MRoIB improves the performance of the MapReduce job running on 8 slaves nodes, by 28–30 %, as compared to default Hadoop MapReduce over IPoIB (56 Gbps). Similarly, Fig. 8(b) illustrates a comparison between MRoIB and default MapReduce over IPoIB (56 Gbps) with 16 slave nodes on Cluster B. It is clear that RDMA-enhanced MapReduce outperforms IPoIB (56 Gbps) by about 25–28 %, even on a larger cluster. This points us towards the benefits that native IB-based MapReduce has over default Hadoop MapReduce running over IPoIB (56 Gbps).

7 Conclusion and Future Work

In order to obtain optimal performance, it is essential to study the impact of network on the performance of Hadoop MapReduce. In this paper, we have designed a micro-benchmark suite to evaluate the performance of stand-alone MapReduce over different network interconnects. This comprehensive and easy-to-use micro-benchmark suite, that is compatible with both Hadoop 1.x and Hadoop 2.x, gives users a means to understand how factors such as intermediate data patterns, size and number of key/value pairs, data type, and number of map and reduce tasks, can influence the execution of a MapReduce job on high-performance networks.

As an illustration, we have presented performance results of Hadoop MapReduce with our micro-benchmarks over different networks/protocols: 1 GigE, 10 GigE, IPoIB QDR (32 Gbps), and IPoIB FDR (56 Gbps). We observe that the performance of the MapReduce job improves around 17 %, if the underlying interconnect is changed to 10 GigE from 1 GigE, and up to 23 %, when changed to IPoIB QDR (32 Gbps). Additionally, IPoIB QDR (32 Gbps) improves performance of the MapReduce job by about 12 % over 10 GigE. It is also noticeable that IPoIB QDR (32 Gbps) performs better with increasing shuffle data sizes. We also present a case study undertaken to understand the benefits that native InfiniBand can provide to Hadoop MapReduce. It is clear that RDMA-enhanced MapReduce design can achieve much better performance than default Hadoop MapReduce over IPoIB FDR (56 Gbps).

In the light of the results presented in this paper, our proposed micro-benchmark suite can help developers enhance their MapReduce designs, especially those intended to optimize data shuffling over the network. For future work, we plan to provide public access to these micro-benchmarks, by making them available as a part of the OSU HiBD Micro-benchmarks [2]. We also intend to add additional features to enhance this micro-benchmark suite, so that users can gain a more concrete understanding of real-world workloads.

References

1. BigDataBench: A Big Data Benchmark Suite. http://prof.ict.ac.cn/BigDataBench
2. High-Performance Big Data (HiBD). http://hibd.cse.ohio-state.edu
3. NullOutputFormat (Hadoop 1.2.1 API). https://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/mapred/lib/NullOutputFormat.html
4. TPC Benchmark H - Standard Specication. http://www.tpc.org/tpch
5. Apache Hadoop NextGen MapReduce (YARN). http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
6. Bennett, C., Grossman, R.L., Locke, D., Seidman, J., Vejcik, S.: Malstone: Towards a benchmark for analytics on large data clouds. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, Washington, DC, USA (2010)

7. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC, Indianapolis, Indiana, USA (2010)
8. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design and Implementation, OSDI, San Francisco, CA (2004)
9. Huang, S., Huang, J., Dai, J., Xie, T., Huang, B.: The HiBench benchmark suite: characterization of the MapReduce-based data analysis. In: Proceedings of the 26th International Conference on Data Engineering Workshops, ICDEW, Long Beach, CA, USA (2010)
10. Islam, N.S., Lu, X., Wasi-ur-Rahman, M., Jose, J., (DK) Panda, D.K.: A micro-benchmark suite for evaluating HDFS operations on modern clusters. In: Rabl, T., Poess, M., Baru, C., Jacobsen, H.-A. (eds.) WBDB 2012. LNCS, vol. 8163, pp. 129–147. Springer, Heidelberg (2014)
11. Islam, N.S., Rahman, M.W., Jose, J., Rajachandrasekar, R., Wang, H., Subramoni, H., Murthy, C., Panda, D.K.: High performance RDMA-based design of HDFS over InfiniBand. In: The International Conference for High Performance Computing, Networking, Storage and Analysis (SC), November 2012
12. Islam, N.S., Lu, X., Rahman, M.W., Panda, D.K.D.: SOR-HDFS: a SEDA-based approach to maximize overlapping in RDMA-enhanced HDFS. In: Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing, HPDC ’14, Vancouver, BC, Canada, pp. 261–264. ACM (2014)
13. Kim, K., Jeon, K., Han, H., Kim, S., Jung, H., Yeom, H.: MRBench: a benchmark for MapReduce framework. In: Proceedings of the IEEE 14th International Conference on Parallel and Distributed Systems, ICPADS, Melbourne, Victoria, Australia (2008)
14. Liang, F., Feng, C., Lu, X., Xu, Z.: Performance benefits of DataMPI: a case study with BigDataBench. In: The 4th Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware, BPOE-4, Salt lake, Utah (2014)
15. Lu, X., Islam, N.S., Rahman, M.W., Jose, J., Subramoni, H., Wang, H., Panda, D.K.: High-performance design of hadoop RPC with RDMA over InfiniBand. In: Proceedings of the IEEE 42th International Conference on Parallel Processing, ICPP, Lyon, France (2013)
16. Lu, X., Islam, N.S., Wasi-Ur-Rahman, M., Panda, D.K.: A Micro-benchmark suite for evaluating hadoop RPC on high-performance networks. In: Proceedings of the 3rd Workshop on Big Data Benchmarking, WBDB (2013)
17. Lu, X., Wang, B., Zha, L., Xu, Z.: Can MPI benefit hadoop and MapReduce applications? In: Proceedings of the IEEE 40th International Conference on Parallel Processing Workshops, ICPPW (2011)
18. Patil, S., Polte, M., Ren, K., Tantisiriroj, W., Xiao, L., López, J., Gibson, G., Fuchs, A., Rinaldi, B.: YCSB++: benchmarking and performance debugging advanced features in scalable table stores. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, SoCC, Cascais, Portugal (2011)
19. Rahman, M.W., Islam, N.S., Lu, X., Jose, J., Subramoni, H., Wang, H., Panda, D.K.: High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand. In: Proceedings of the IEEE 27th International Symposium on Parallel and Distributed Processing Workshops and PhD Forum. IPDPSW, Washington, DC, USA (2013)

20. Rahman, M.W., Lu, X., Islam, N.S., Panda, D.K.: HOMR: a hybrid approach to exploit maximum overlapping in MapReduce over high performance interconnects. In: Proceedings of the 28th ACM International Conference on Supercomputing, ICS '14, Munich, Germany, pp. 33–42. ACM (2014)
21. Sangroya, A., Serrano, D., Bouchenak, S.: MRBS: towards dependability benchmarking for hadoop MapReduce. In: Caragiannis, I., et al. (eds.) Euro-Par 2012 Workshops 2012. LNCS, vol. 7640, pp. 3–12. Springer, Heidelberg (2013)
22. Stampede at Texas Advanced Computing Center. http://www.tacc.utexas.edu/resources/hpc/stampede
23. The Apache Software Foundation: Apache Hadoop. http://hadoop.apache.org
24. Top500 Supercomputing System. http://www.top500.org
25. Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., Gao, W., Jia, Z., Shi, Y., Zhang, S., Zheng, C., Lu, G., Zhan, K., Li, X., Qiu, B.: BigDataBench: a big data benchmark suite from internet services. In: Proceedings of the 20th IEEE International Symposium on High Performance Computer Architecture, HPCA, Orlando, Florida (2014)

http://www.springer.com/978-3-319-13020-0

Big Data Benchmarks, Performance Optimization, and
Emerging Hardware
4th and 5th Workshops, BPOE 2014, Salt Lake City, USA,
March 1, 2014 and Hangzhou, China, September 5, 2014,
Revised Selected Papers
Zhan, J.; Han, R.; Weng, C. (Eds.)
2014, X, 221 p. 100 illus., Softcover
ISBN: 978-3-319-13020-0